





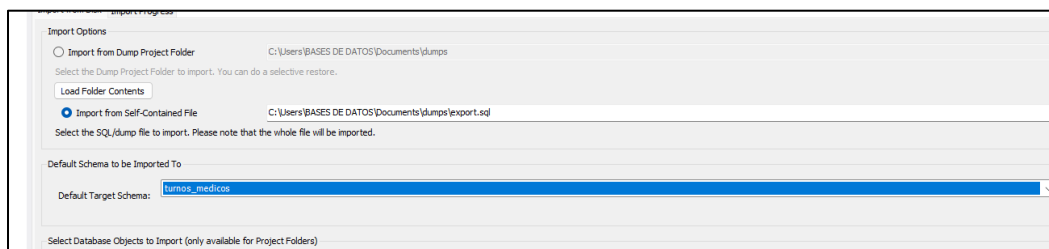
TALLER Sistema con Base de Datos y Java

Parte 1: Configuración de la base de datos

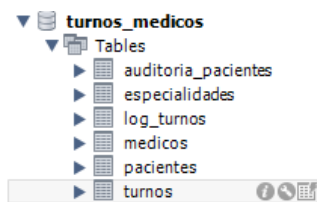
1. Descargar el script SQL proporcionado.

	Taller comprensión de BD	2/7/2025 11:10	Documento de Mi...	28 KB
	Taller 2	2/7/2025 11:10	Documento de Mi...	28 KB
	Script base medica	2/7/2025 11:10	SQL Text File	6 KB
	TurnosMedicosSoloCodigo	2/7/2025 11:15	Carpeta de archivos	

2. Crear la base de datos y las tablas ejecutando el script.



3. Insertar los registros necesarios utilizando INSERT INTO.



Script para inserts

```
-- Especialidades
INSERT INTO especialidades (nombre) VALUES
('Gastroenterología'),
('Endocrinología'),
('Oftalmología'),
('Urología'),
('Oncología'),
('Psiquiatría'),
('Reumatología'),
('Otorrinolaringología'),
('Nefrología'),
('Geriatría'),
('Infectología'),
('Cirugía General'),
('Traumatología'),
('Neumología'),
('Hematología'),
```

```

('Alergología');

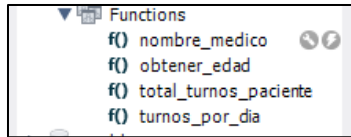
-- Médicos
INSERT INTO medicos (nombre, especialidad_id) VALUES
('Dr. Mario Castro', 5),
('Dra. Paula Vega', 6),
('Dr. Andrés Silva', 7),
('Dra. Carla Ramírez', 8),
('Dr. Roberto Méndez', 9),
('Dra. Isabel Ortega', 10),
('Dr. Hugo Fernández', 11),
('Dra. Valeria Ríos', 12),
('Dr. Julio Salazar', 13),
('Dra. Beatriz Cedeño', 14),
('Dr. Esteban Mora', 15),
('Dra. Fabiola Lema', 16),
('Dr. Manuel Paredes', 17),
('Dra. Gabriela Núñez', 18),
('Dr. Ramiro Vargas', 19),
('Dra. Sonia Gómez', 20);

-- Pacientes
INSERT INTO pacientes (nombre, cedula, fecha_nacimiento) VALUES
('Jorge Castillo', '1106677889', '1975-03-14'),
('Andrea Viteri', '1107788990', '1992-07-22'),
('Patricio Lema', '1108899001', '1980-11-30'),
('Sofía Paredes', '1109900112', '2001-02-08'),
('Fernando Mora', '1110011223', '1968-05-17'),
('Lorena Ríos', '1111122334', '1999-09-25'),
('Hugo Andrade', '1112233445', '1973-01-05'),
('Paula Méndez', '1113344556', '1987-04-12'),
('Esteban Cedeño', '1114455667', '1994-06-28'),
('Gabriela Salazar', '1115566778', '1983-08-03'),
('Ricardo Núñez', '1116677889', '2000-10-11'),
('Sonia Vargas', '1117788990', '1977-12-19'),
('Ramón Gómez', '1118899001', '1965-03-21'),
('Natalia Torres', '1119900112', '1996-07-15'),
('Pablo Herrera', '1120011223', '1989-05-27'),
('Luciana Ortega', '1121122334', '2003-11-09');

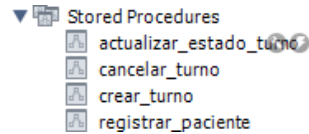
-- Turnos
INSERT INTO turnos (paciente_id, medico_id, fecha, hora) VALUES
(5, 1, CURDATE() + INTERVAL 2 DAY, '08:30:00'),
(6, 2, CURDATE() + INTERVAL 3 DAY, '09:00:00'),
(7, 3, CURDATE() + INTERVAL 4 DAY, '11:00:00'),
(8, 4, CURDATE() + INTERVAL 5 DAY, '14:00:00'),
(9, 5, CURDATE() + INTERVAL 6 DAY, '10:30:00'),
(10, 6, CURDATE() + INTERVAL 7 DAY, '13:00:00'),
(11, 7, CURDATE() + INTERVAL 8 DAY, '15:00:00'),
(12, 8, CURDATE() + INTERVAL 9 DAY, '09:30:00'),
(13, 9, CURDATE() + INTERVAL 10 DAY, '11:30:00'),
(14, 10, CURDATE() + INTERVAL 11 DAY, '12:00:00'),
(15, 11, CURDATE() + INTERVAL 12 DAY, '08:00:00'),
(16, 12, CURDATE() + INTERVAL 13 DAY, '10:00:00'),
(17, 13, CURDATE() + INTERVAL 14 DAY, '09:15:00'),
(18, 14, CURDATE() + INTERVAL 15 DAY, '11:45:00'),
(19, 15, CURDATE() + INTERVAL 16 DAY, '14:30:00'),
(20, 16, CURDATE() + INTERVAL 17 DAY, '10:45:00');

```

4. Crear únicamente las funciones definidas en el script.



5. Crear únicamente los procedimientos almacenados.



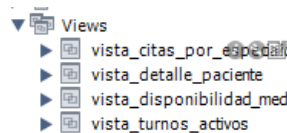
6. Crear únicamente los triggers.

6.1 Verificar que los triggers fueron creados correctamente usando la instrucción:

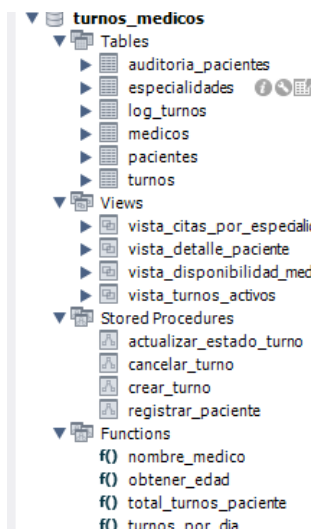
SHOW TRIGGERS;

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
trg_validar_paciente_nuevo	INSERT	pacientes	BEGIN INSERT INTO auditoria_pacientes(pac...	AFTER	2025-07-02 11:12:06.92	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb3	utf8mb3_general_ci	utf8mb4_0900_ai_ci
trg_validar_fecha_turno	INSERT	turnos	BEGIN IF NEW.fecha < CURDATE() THEN ...	BEFORE	2025-07-02 11:12:06.89	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb3	utf8mb3_general_ci	utf8mb4_0900_ai_ci
trg_log_cambios_turnos	UPDATE	turnos	BEGIN INSERT INTO log_turnos(turno_id, acc...	AFTER	2025-07-02 11:12:06.91	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb3	utf8mb3_general_ci	utf8mb4_0900_ai_ci

7. Crear las vistas definidas.

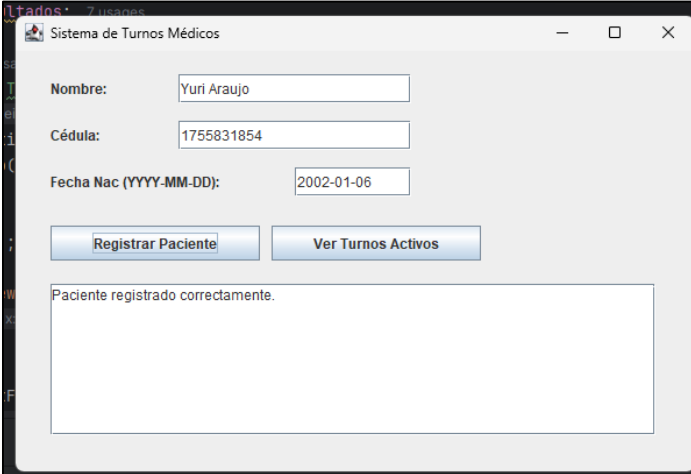


8. Verificar que todos los elementos anteriores (tablas, funciones, procedimientos, triggers y vistas) fueron creados correctamente.



Parte 2: Configuración del proyecto Java en IntelliJ IDEA

1. Abrir IntelliJ IDEA y cargar el proyecto.
2. Restaurar el código Java del sistema.
3. Ir a File → Project Structure → Libraries y añadir el conector MySQL (mysql-connector-java-x.x.xx.jar) previamente descargado.
4. Si aún no lo tienes, descargar el conector MySQL desde el sitio oficial:
<https://dev.mysql.com/downloads/connector/j/>
5. Ejecutar el código Java y comprobar que la conexión con la base de datos funcione.



Sistema de Turnos Médicos

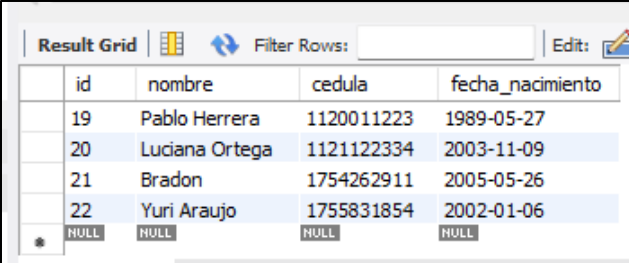
Nombre: Yuri Araujo

Cédula: 1755831854

Fecha Nac (YYYY-MM-DD): 2002-01-06

Registrar Paciente Ver Turnos Activos

Paciente registrado correctamente.



	id	nombre	cedula	fecha_nacimiento
	19	Pablo Herrera	1120011223	1989-05-27
	20	Luciana Ortega	1121122334	2003-11-09
	21	Bradon	1754262911	2005-05-26
	22	Yuri Araujo	1755831854	2002-01-06
*	NULL	NULL	NULL	NULL

6. Verificar que el código Java utilice correctamente los elementos de base de datos:

- Vistas

```

private void mostrarTurnos() { 1usage
    try (Connection conn = DBConnection.getConnection()) {
        String query = "SELECT * FROM vista_turnos_activos";
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();
        StringBuilder sb = new StringBuilder();
        while (rs.next()) {
            sb.append("ID: ").append(rs.getInt( columnLabel: "id")).append(" - ")
              .append("Paciente: ").append(rs.getString( columnLabel: "paciente")).append(" - ")
              .append("Médico: ").append(rs.getString( columnLabel: "medico")).append(" - ")
              .append("Fecha: ").append(rs.getDate( columnLabel: "fecha")).append("\n");
        }
        txtResultados.setText(sb.toString());
    } catch (SQLException ex) {
        txtResultados.setText("Error: " + ex.getMessage());
    }
}

```

Sistema de Turnos Médicos

Nombre:

Cédula:

Fecha Nac (YYYY-MM-DD):

ID: 14 - Paciente: Gabriela Salazar - Médico: Dra. Isabel Ortega - Fecha: 2025-07-13
 ID: 15 - Paciente: Ricardo Núñez - Médico: Dr. Hugo Fernández - Fecha: 2025-07-14
 ID: 16 - Paciente: Sonia Vargas - Médico: Dra. Valeria Ríos - Fecha: 2025-07-15
 ID: 17 - Paciente: Ramón Gómez - Médico: Dr. Julio Salazar - Fecha: 2025-07-16
 ID: 18 - Paciente: Natalia Torres - Médico: Dra. Beatriz Cedeño - Fecha: 2025-07-17
 ID: 19 - Paciente: Pablo Herrera - Médico: Dr. Esteban Mora - Fecha: 2025-07-18
 ID: 20 - Paciente: Luciana Ortega - Médico: Dra. Fabiola Lema - Fecha: 2025-07-19

Pdta. No muestra los registros que modifique con el procedimiento almacenado en activo porque la vista está diseñada para mostrar los de estado 'pendiente', más no los de activo, para muestra, retornaré una de las citas al estado pendiente y se mostrará en el Java:

Call stored procedure turnos_medicos.actualizar_estado_t...

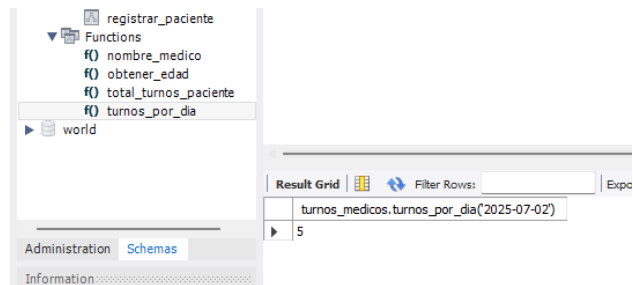
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

t_id 22 [IN] INT

nuevo_estado pendiente [IN] VARCHAR(20)

ID: 15 -	Paciente: Ricardo Núñez -	Médico: Dr. Hugo Fernández -	Fecha: 2025-07-14
ID: 16 -	Paciente: Sonia Vargas -	Médico: Dra. Valeria Ríos -	Fecha: 2025-07-15
ID: 17 -	Paciente: Ramón Gómez -	Médico: Dr. Julio Salazar -	Fecha: 2025-07-16
ID: 18 -	Paciente: Natalia Torres -	Médico: Dra. Beatriz Cedeño -	Fecha: 2025-07-17
ID: 19 -	Paciente: Pablo Herrera -	Médico: Dr. Esteban Mora -	Fecha: 2025-07-18
ID: 20 -	Paciente: Luciana Ortega -	Médico: Dra. Fabiola Lema -	Fecha: 2025-07-19
ID: 22 -	Paciente: Bradon -	Médico: Dra. Carla Gómez -	Fecha: 2025-07-02

- Funciones



The screenshot shows a database interface with a tree view on the left and a result grid on the right.

Tree View:

- registrar_paciente
 - Functions
 - f() nombre_medico
 - f() obtener_edad
 - f() total_turnos_paciente
 - f() turnos_por_dia
 - world

Result Grid:

turnos_medicos.turnos_por_dia('2025-07-02')
5

- Procedimientos almacenados

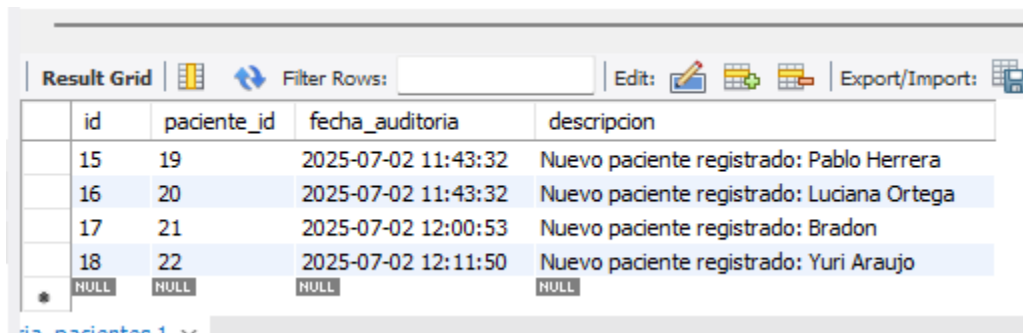
Call stored procedure turnos_medicos.crear_turno

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_id	21	[IN]	INT
m_id	3	[IN]	INT
f	2025-07-02	[IN]	DATE
h	14:00:00	[IN]	TIME

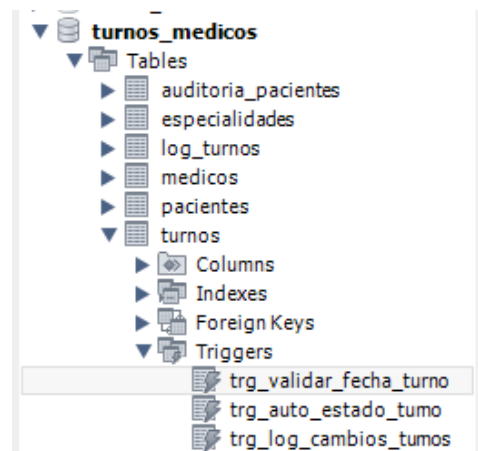
Execute Cancel

- Triggers

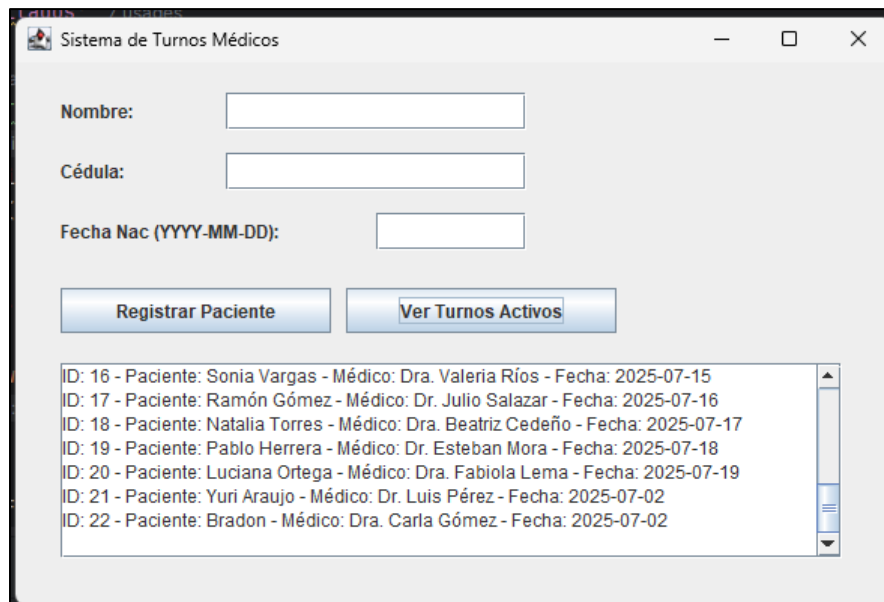


The screenshot shows a database management interface with a 'Result Grid' containing patient registration data. The grid has columns for 'id', 'paciente_id', 'fecha_auditoria', and 'descripcion'. The data rows show new patient registrations with IDs 15 through 18. A summary row at the bottom shows 'NULL' values for the first three columns and 'NULL' for the description.

	id	paciente_id	fecha_auditoria	descripcion
	15	19	2025-07-02 11:43:32	Nuevo paciente registrado: Pablo Herrera
	16	20	2025-07-02 11:43:32	Nuevo paciente registrado: Luciana Ortega
	17	21	2025-07-02 12:00:53	Nuevo paciente registrado: Bradon
	18	22	2025-07-02 12:11:50	Nuevo paciente registrado: Yuri Araujo
*	NULL	NULL	NULL	NULL



7. Crear usuarios para probar las acciones del trigger



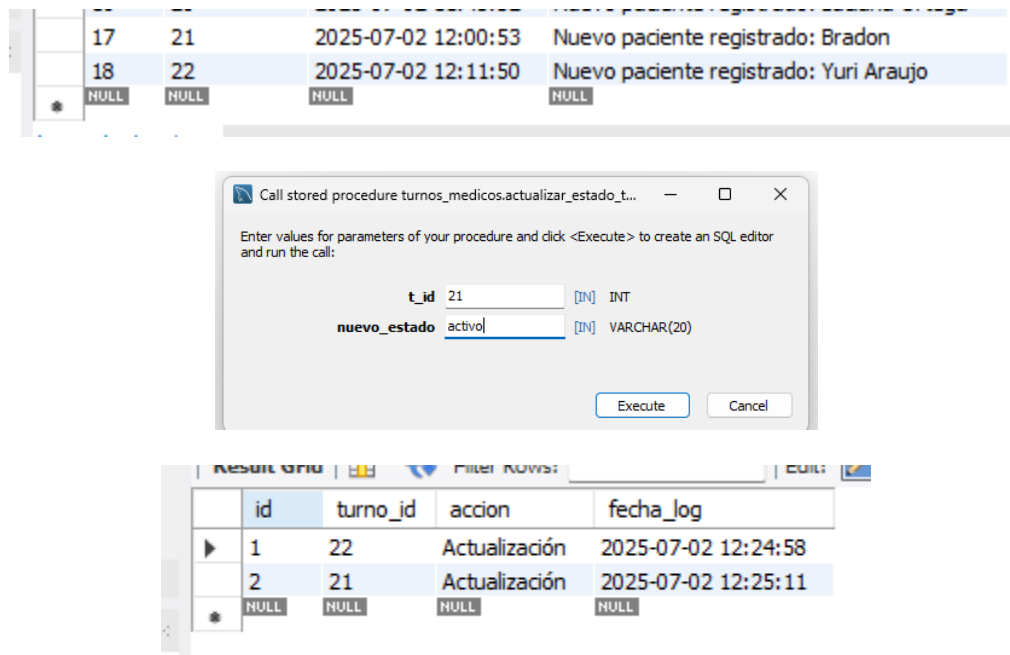
The screenshot shows a window titled 'Sistema de Turnos Médicos'. It contains a form for registering a patient with fields for 'Nombre:', 'Cédula:', and 'Fecha Nac (YYYY-MM-DD):'. Below the form are two buttons: 'Registrar Paciente' and 'Ver Turnos Activos'. At the bottom of the window is a list box showing a list of active appointments, including patient names, doctor names, and dates.

Nombre:

Cédula:

Fecha Nac (YYYY-MM-DD):

ID: 16 - Paciente: Sonia Vargas - Médico: Dra. Valeria Ríos - Fecha: 2025-07-15
ID: 17 - Paciente: Ramón Gómez - Médico: Dr. Julio Salazar - Fecha: 2025-07-16
ID: 18 - Paciente: Natalia Torres - Médico: Dra. Beatriz Cedeño - Fecha: 2025-07-17
ID: 19 - Paciente: Pablo Herrera - Médico: Dr. Esteban Mora - Fecha: 2025-07-18
ID: 20 - Paciente: Luciana Ortega - Médico: Dra. Fabiola Lema - Fecha: 2025-07-19
ID: 21 - Paciente: Yuri Araujo - Médico: Dr. Luis Pérez - Fecha: 2025-07-02
ID: 22 - Paciente: Bradon - Médico: Dra. Carla Gómez - Fecha: 2025-07-02



GITHUB: [brandonvht26/Taller_conexion_BDD](https://github.com/brandonvht26/Taller_conexion_BDD)

Conclusión: El emparejamiento de la base de datos con el programa en su fase back es una de las implementaciones más importantes en el desarrollo de software, porque así es como realmente un programa real interactúa, en base a una base de datos de la cual es capaz de sustraer dicha información y presentarla al usuario. Depende el tipo de usuario se han de generar diversos tipos de vistas y también es justamente mediante la creación de vistas, procedimientos almacenados y funciones que se reduce la tarea del backend porque se puede gestionar la base de datos y simplemente consumirla y automatizarla desde el backend. Esta práctica es muy útil en el contexto de un desarrollador backend, para comprender como emparejar una base de datos con un programa y que con una conexión de frontend, pueda el usuario consumir la información en una interfaz simple e intuitiva.